(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 June 2001 (14.06.2001)

PCT

(10) International Publication Number
WO 01/42881 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US00/42665

(22) International Filing Date: 6 December 2000 (06.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/169,101 6 December 1999 (06.12.1999) US
Not furnished 5 December 2000 (05.12.2000) US

(71) Applicant: B-BOP ASSOCIATES, INC. [US/US]; Suite 100, One Bay Plaza, 1350 Old Bayshore Highway, Burlingame, CA 94010 (US).

(72) Inventors: DODDS, David; 16 Old Barn Road, Stamford, CT 06905 (US). KUO, Larry; 120 Morning Star Drive, San Jose, CA 95131 (US). SENGUPTA, Soumitra; 15 First Street, Apt. 5, Stamford, CT 06905 (US). LINDSEY, Bill; 2203 Hastings Drive, Apt. 28, Belmont, CA 94002 (US).

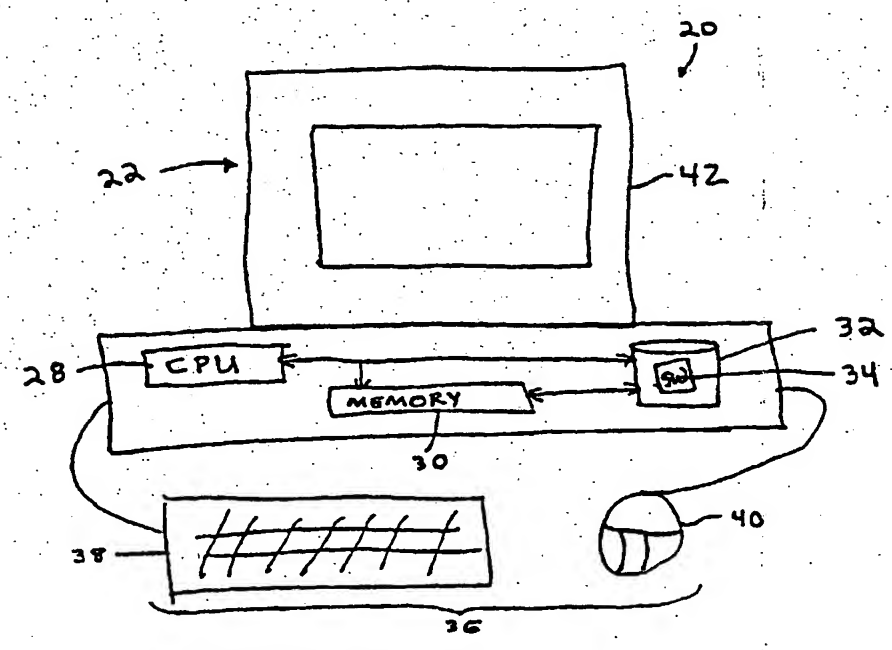
(74) Agent: LOHSE, Timothy, W.; Gray Cary Ware & Freidenrich LLP, 400 Hamilton Avenue, Palo Alto, CA 94301-1825 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR THE STORAGE, INDEXING AND RETRIEVAL OF XML DOCUMENTS USING RELATIONAL DATABASES



(57) Abstract: A system and method for assigning attributes to XML document nodes to facilitate their storage in relational databases and the subsequent retrieval and reconstruction of pertinent nodes and fragments in original document order is provided. Since these queries are performed using relational database query engines, the speed of their execution is significantly faster than that using more exotic systems such as object-oriented databases. Furthermore, this method is portable across all vendor platforms, and so can be deployed at client sites without additional investments in database software.

WO 01/42881 A2

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR THE STORAGE, INDEXING AND RETRIEVAL OF XML DOCUMENTS USING RELATIONAL DATABASES

Priority Claim

This application claims priority under 35 USC §§ 119 and 120 from US Provisional Patent Application No. 60/169,101 filed December 6, 1999.

Background of the Invention

This invention relates generally to a system and method for storing documents in one format in a database having a different format and in particular to a system and method for storing and retrieving eXtensible Markup Language (XML) documents using a relational database.

The new eXtensible Markup Language (XML) protocol is poised to become the lingua franca of the Internet for capturing and electronically transmitting information. The advantage of XML, as compared to the older hypertext markup language protocol (HTML), is that it contains tags which render semantic significance to the information between the tags (e.g., the text between the tags is the last name of an author). In contrast, HTML tags are used primarily for specifying how the information is to be displayed in a browser (e.g., show the text between the tags in bold Arial font). Additionally, using known eXtensible Stylesheets (written in XSL), one may specify not only the format of how different XML elements are to be shown in a browser, but also the order in which they are to be displayed. These features of XML give a user much greater power and flexibility in searching for relevant information since a search may be performed using the tags that contain the semantic information. In addition, XML permits examining the information from different perspectives once it is found by the user.

To take full advantage of the possibilities that the XML protocol affords, it is desirable to devise an efficient means of storing, indexing and retrieving (via queries) XML documents. Typical RDMS, ODMS and flat files are slow and inefficient at storing XML documents. A preferred way of building Document Object Model (DOM) representations of the XML documents and then traversing the resulting trees to locate relevant nodes is only acceptable for small documents since memory becomes a limiting factor when the XML documents approach even moderate sizes. In addition, searches are not optimal since all searches must begin at the root of the document instead of at any node in the document. Moreover, it is not possible to search across a collection of documents (e.g. poems, novels, short stories and plays) for a particular character or the author.

At the same time, XML documents present unique challenges to storage in relational databases since their semi-structured nature often leads to a proliferation of tables when normalization is carried out. Given that relational database technology has seen great strides over the past couple of decades, it would be desirable and useful to provide a clean way of representing XML documents in relational terms. It is therefore the goal of the present invention to provide a system and method for the storage, indexing and retrieval of XML documents using relational databases.

Summary of the Invention

A system and method for storing, indexing and retrieving XML documents in a relational database is provided in accordance with the invention. The method may include identifying and assigning properties and encodings to the nodes of an XML document that will make them amenable to storage and retrieval using relational

databases. The method has several advantages. It allows the system to capture and reproduce the structure of not only the whole document, but fragments of each document as well. It also permits a user to traverse the XML tree, figuratively, by means of string manipulation queries instead of following pointers in memory or computing joins between tables, which are computationally more expensive operations. Finally, the properties and encodings that are attached to the nodes are compact and can be effectively indexed, thus enhancing the performance of queries against the database.

The system in accordance with the invention uses any relational database management system to store the XML documents so that the system and method are not dependent on any particular relational database implementation. The system permits a user to search through the XML documents stored in the relational database from any node element without starting from the root element of the document. This provides optimal efficiency during search and retrieval that can not be obtained using other methods today. In addition, a document may be constructed from any node and its descendants. The system also permits documents conforming to any XML schema to be stored in an efficient manner. The system can also store any well formed XML document that do not conform to any schema or DTD (Document Type Definition). This is an important feature as a large majority of XML documents generated do not conform to a schema or DTD.

In accordance with the invention, the system may include a converter and a searcher that permit XML documents to be stored in the relational database and retrieved from a relational database using typical SQL queries. In a preferred embodiment, the converter and searcher may be one or more software modules being executed by a central processing unit on a computer system. In accordance with the invention, the method for storing the XML documents may include the steps of generating an

XMLName value for each element in the document tree, generating a NamePath value for each node of the document and generating an OrderPath value for each node of the document. Collectively, assigning values to these elements are called encodings. These encodings result in efficient storage, indexing and searching of XML documents without destroying the underlying hierarchical structure of the documents. The retrieval of the XML documents once they are in the relational database is relatively easy since typical string matching SQL queries may be used.

Thus, in accordance with the invention, a computer system and method for manipulating an XML document using a relational database is provided. The system comprises a converter that receives an XML document and generates a set relational database tables based on the hierarchical structure of XML a database for storing the relational database tables, and a searcher for querying the generated relational database table in the database to locate content originally in the XML document that is now stored in the relational database tables wherein the located content is returned to the user as an XML document or a portion of an XML document as desired by the user which can be another software module. The invention also includes the searcher that can convert queries specified on the XML document or document collections and convert them to simple SQL queries to retrieve the content desired by the user.

In accordance with another aspect of the invention, a computer system for storing an XML document using a relational database is provided wherein the system comprises a converter that receives an XML document and generates relational database tables based on the structure of the XML document. The converter further comprises a software module that generates a unique name attribute for each node in the XML document, a software module that generates a path attribute for a particular node of the XML document wherein the path attribute comprises a list of the name attributes for the one or more nodes from the particular node to a root node of the XML document, a software module that generates an order attribute for the particular node,

the order attribute comprising an enumerated order of the particular node from the root node to the particular node, and a software module that generates a NodeValue attribute containing a value of the particular node. Collectively these attributes are called encodings that result in efficient storage, indexing and searching of XML documents without destroying the underlying hierarchical structure of the documents.

In accordance with yet another aspect of the invention, a data structure that stores a node of interest of an XML document in a relational database is provided. The data structure comprises an XMLName attribute comprising a unique name for the node of interest, a NamePath attribute comprising a list of the XMLName attributes for the one or more nodes from the node of interest to a root node of the XML document, an OrderPath attribute comprising an enumerated order of the node of interest from the root node to the node of interest, and a NodeValue attribute containing a value of the node of interest. Collectively these attributes are called encodings that result in efficient storage, indexing and searching of XML documents without destroying the underlying hierarchical structure of the documents.

Brief Description of the Drawings

Figure 1 is a diagram illustrating a personal computer implementation of an XML document storage and retrieval system in accordance with the invention;

Figure 2 is a diagram illustrating more details of the XML document storage and retrieval system in accordance with the invention;

Figure 3 is a diagram illustrating an example of a document type definition (DTD) tree for an XML document;

Figure 4 is a diagram illustrating an XML document corresponding to the table shown in Figure 3;

Figure 5 is a flowchart illustrating an example of a method for storing XML documents in a relational database in accordance with the invention; and

Figure 6 is a flowchart illustrating a method for retrieving an XML document from a search of a relational database in accordance with the invention.

Detailed Description of a Preferred Embodiment

The invention is particularly applicable to a software implemented XML document storage and retrieval system and method and it is in this context that the invention will be described. It will be appreciated, however, that the system and method in accordance with the invention has greater utility since it may be implemented in hardware instead of software.

Figure 1 is a block diagram illustrating an embodiment of a software-based XML document storage and retrieval system 20 in accordance with the invention. In this embodiment, the storage and retrieval system 20 may be executed by a computer 22. The computer 22 may be a typical stand-alone personal computer, a computer connected to a network, a client computer connected to a server or any other suitable computer system. For purposes of illustration only, an embodiment using a stand-alone computer 22 will be described herein.

The computer 22 may include a central processing unit (CPU) 28, a memory 30, a persistent storage device 32, such as a hard disk drive, a tape drive, an optical drive or the like and a storage and retrieval system 34. In a preferred embodiment, the storage and retrieval system may be one or more software applications stored in the persistent storage device 32 of the computer that may be loaded into the memory 30 so that the storage and/or retrieval functionality of the storage and retrieval system may be executed by the CPU 28. The computer 22 may be connected to a remote server or

other computer networks that permit the computer 22 to network with and share the stored XML document with other computers or to perform searches on XML stored documents on other computer systems.

The computer 22 may further include one or more input devices 36, such as a keyboard 38, a mouse 40, a joystick or the like, a display 42 such as a typical cathode ray tube, a flat panel display or the like and one or more output devices (not shown) such as a printer for producing printed output of the search results. The input and output devices permit a user of the computer to interact with the storage and retrieval system so that the user may, for example, enter a query using the input devices and view the results of the query on the display or print the query results.

As described below in more detail, the storage and retrieval system 34 may include one or more different software modules that provide XML document storage capabilities and XML document retrieval capabilities in accordance with the invention. Now, more details of the storage and retrieval system will be described.

Figure 2 is a diagram illustrating more details of the XML document storage and retrieval system 34 in accordance with the invention. The system may include a converter module 50, a searcher module 52 and a relational database 54. Each of the modules may be implemented, in a preferred embodiment, as a software application being executed by a CPU as described above. The relational database 54 may be any type of relational database so that the system 34 in accordance with the invention may be used to store XML documents in any relational database system.

The converter module 50 accepts XML documents, processes them and outputs relational data about the XML documents as described below that is stored in the typical relational database 54. The searcher module 52 generates a user interface to a user, permits the user to enter a text string type relational database query, processes the

query by communicating a query to the relational database 54 and sends the results of the query in its original XML form to the user so that the user may view or print the query results. In combination, the two modules shown permit XML documents to be stored in any relational database system and then permits a user to enter a typical text string relational database query in order to retrieve XML documents from the relational database that match the text string query. Each of these modules will be described in more detail below. Now, an example of a Document Type Definition (DTD) of an XML document will be described to better understand the invention. This example of the DTD will be used as an example to illustrate the storage and retrieval system in accordance with the invention.

Figure 3 is a diagram illustrating an example of a Document Type Definition (DTD) tree 60 for an XML document. Although not required to do so, an XML document typically conforms to a DTD which, loosely speaking, is a schema for the data found in the document. However, XML documents are semi-structured in the sense that there are elements specified in the DTD that may be optionally present and some that may be present more than once. This is in contrast to typical relational database tables where each record must have either zero (if it is NULL) or only one value for an attribute.

XML documents also resemble an object-oriented database in that there are parent-child relationships between elements which are not found between attributes in a relational database. The following example of an XML document should help make these distinctions more clear. An example of the XML DTD syntax may be:

```
<!ELEMENT library (book*, periodical*)>
<!ELEMENT book (title, author+)>
<!ATTLIST book edition CDATA #REQUIRED>
```

<!ELEMENT author (title?, firstname, lastname)>

In the above example, elements that appear within parentheses are the children of elements before the parentheses. In addition a "*" denotes 0 or more occurrences of the element, a "+" denotes one or more occurrences and a "?" denotes 0 or 1 occurrence. The above example DTD may be represented by the DTD tree shown in Figure 3. The DTD tree 60 may include a root node 62 (containing the element "library" in this example), one or more intermediate nodes 64 and one or more leaf nodes 66 that do not have any further nodes attached to them. An example of an XML document 70 that conforms to the DTD is shown in Figure 4. It contains the instances of elements in the DTD tree along with data for each element. The conversion of this example of an XML document into a format that may be stored in a relational database in accordance with the invention will now be described.

Figure 5 is a flowchart illustrating an example of a method 80 for storing XML documents in a relational database in accordance with the invention. The method involves computing three properties, each of which is described below, for each XML document node so that the XML document may be stored, in an efficient manner, in a relational database. The encoding scheme set forth below is a preferred encoding embodiment. However, other encoding schemes may also be used. For example, the encoding set forth below (e.g., 1/2/5/6) may be represented as 1 raised to the power 1, 2 raised to the power 2, 3 raised to the power 5 and 4 raised to the power 6 and so on. That way, instead of performing string manipulation, the system would be doing factorization. Based on this other encoding, the factorization approach can generate faster queries and save indexing and database space. Thus, the invention is not limited to any particular encoding and the encodings in accordance with the invention are

created based on the structure of the document and then the encodings are used to store, index and search for the content while preserving the hierarchy of the document. In a first step 81 of the method, it is determined if an element is ready for processing. If there is an element ready for processing, then the method generates an XMLName property for the particular element. If an element is not ready for processing, but an attribute of the XML document is read for processing, then the method also generates the XMLName property for the particular attribute. In more detail, the method starts by assigning each element name a unique XMLName property (in this example, the property is alphanumeric). For the example above, we could assign the XMLNames as shown in Table 1 (the XMLName Table).

Table 1 (the "XMLName Table")

Element or Attribute Name	XMLName
library	1
book	2
periodical	3
edition	4
title	5
author	6
firstname	7
lastname	8

Note that "title" gets only one XMLName value even though the element appears twice in the DTD tree as either the title of a book or the title of an author. This allows for more XMLName attributes to be encoded given strings of a specific length.

Now, in step 84, a NamePath value is automatically determined for each node of the DTD tree. In particular, the NamePath value may be constructed from the XMLNames of each node on the path from the root node to the node of interest. From this analysis, we obtain the following table of NamePath values for the example XML document:

NamePath Table

DTD Node	NamePath
library	1
library/book	1/2
library/periodical	1/3
library/book/edition	1/2/4
library/book/title	1/2/5
library/book/author	1/2/6
library/book/author/title	1/2/6/5
library/book/author/firstname	1/2/6/7
library/book/author/lastname	1/2/6/8

As shown in the table, each DTD node, such as "library/book/author/lastname ", has a corresponding NamePath value, such as "1/2/6/8". In this manner, using the NamePath values,. it is possible to navigate through the XML document using the relational database. In other words, using this table, the path to any node in the DTD tree (and hence the XML document) may be easily determined. This table may also be stored in the relational database.

Next, in step 86, the method may automatically generate an OrderPath value for each node in the XML document. In particular, each number in the slash-separated OrderPath (see the table below) denotes the breadth-wise enumerated order of the node on the path from the root to the node of interest. Each document node may also inherit the NamePath of the DTD node of which it is an instance. A full DocNode Table for the example XML document looks like this:

DocNode Table

NodeName	NamePath	OrderPath	NodeValue
library	1	1	
book	1/2	1/1	
edition	1/2/4	1/1/1	first
title	1/2/5	1/1/2	The XML Revolution
author	1/2/6	1/1/3	
title	1/2/6/5	1/1/3/1	Software Engineer
firstname	1/2/6/7	1/1/3/2	David
lastname	1/2/6/8	1/1/3/3	Hollenbeck
author	1/2/6	1/1/4	
title	1/2/6/5	1/1/4/1	Chief Architect
firstname	1/2/6/7	1/1/4/2	Carol
lastname	1/2/6/8	1/1/4/3	Bohr
book	1/2	1/2	
edition	1/2/4	1/2/1	second
title	1/2/5	1/2/2	Java Classes for XML
author	1/2/6	1/2/3	
firstname	1/2/6/7	1/2/3/1	Carol
lastname	1/2/6/8	1/2/3/2	Hollenbeck
author	1/2/6	1/2/4	
title	1/2/6/5	1/2/4/1	XML Guru
firstname	1/2/6/7	1/2/4/2	David
lastname	1/2/6/8	1/2/4/3	Bohr

As shown in the Table that may be stored in a relational database, each document node may include a NodeName value (the name of the element), a NamePath value (See above), an OrderPath Value (automatically generated during this step), and a NodeValue value (containing the actual data in that particular node).

In step 88, the method determines if there are any more nodes to process and loops back to step 81 if there are more nodes. If all of the nodes have been processed, then the DocNode Table may be saved in the relational database. In this manner, an XML document is automatically processed in order to generate a DocNode Table that may be stored in any relational database. Once the DocNode table is generated by the system, it may be searched as will now be described in more detail.

Figure 6 is a flowchart illustrating a method 100 for retrieving an XML document from a search of a relational database in accordance with the invention. In step 102, the user or the system using user input, may generate a relational database query. In step 104, the system may query the relational database and in step 106, the query results are output to the user. In accordance with the invention, the system may convert the query results back into references to portions of the XML document so that the user may review the portions of the XML document retrieved during the search in step 108. Now, several examples of retrieving XML documents based on a relational database search will be provided. In particular, a few examples will be shown of how the system may use the NamePath and OrderPath values to select nodes with desired attributes from the XML document repository and also may construct fragments of the original XML documents containing these selected nodes. In all the sample queries below, we assume that we know the context (i.e., the position within the DTD tree) of the nodes we are interested in.

In a first example, a user wants to query the XML document repository to return the titles of all books who have an author with the title of "Chief Architect". Since we know the context of title (i.e., library/book/author/title), we can consult the XMLName Table to obtain the relevant XMLNames and construct the NamePath of title which is "1/2/6/5" in this example. Then, the system may issue the first query that is:

"Select OrderPath from DocNodeTable where NamePath = '1/2/6/5' and NodeValue = 'Chief Architect'"

This query returns an OrderPath of "1/1/4/1" as the result. Since we also know that the element "book" is a grand-parent of element "title", we can deduce that its OrderPath is 1/1. Finally we construct the NamePath of the element "book title" as "1/2/5" and execute the second query that is :

"Select NodeValue from DocNodeTable where NamePath = '1/2/5' and OrderPath like '1/1/%'".

This second query returns the value "The XML Revolution" as the result. This result accomplishes the user goal of returning all books whose author's title is "Chief Architect". In this manner, the XML document repository is queried using typical relational database queries.

In this second example, the user wants to search for the titles of all books who have an author by the name of Carol Hollenbeck. To accomplish this, the system may generate a first query to select the OrderPaths of all firstname nodes with the value Carol:

"Select OrderPath from DocNodeTable where NamePath = '1/2/6/7' and NodeValue = 'Carol'".

This query returns "1/1/4/2" and "1/2/3/1" as the result set. Next, a second query is generated to select the OrderPaths of all lastname nodes with the value Hollenbeck:

"Select OrderPath from DocNodeTable where NamePath = '1/2/6/8' and NodeValue = 'Hollenbeck'"

This query returns "1/1/3/3" and "1/2/3/2" as the result set. Since we know firstname and lastname nodes of the same person belong to the same parent author node, we can deduce from the result sets that only the nodes with OrderPaths "1/2/3/1" and "1/2/3/2" are of interest to us. Thus, we want the title of the book with OrderPath 1/2, which we can retrieve with the following query:

"Select NodeValue from DocNodeTable where NamePath = '1/2/5' and OrderPath like '1/2/%'"

This query returns "Java Classes for XML" as the result which is the proper result.

In a third example, the user wants to be returned all the information pertaining to the authors of "The XML Revolution" and presented in the original document order. Thus, first, the OrderPath of the relevant title node is determined by the following query:

"Select OrderPath from DocNodeTable where NamePath = '1/2/5' and NodeValue = 'The XML Revolution'"

This query returns "1/1/2" as the result. Thus, as a result of the first query, we know that the OrderPath of the relevant book node is "1/1". Since the nodes for all author information are descendants of the author node (that has NamePath "1/2/6"), which in turn is a child of the "book" node, we can execute the following query to obtain the required result:

"Select NodeValue from DocNodeTable where NamePath like '1/2/6/%' and OrderPath like '1/1/%' Order by OrderPath"

This query returns "Software Engineer, David, Hollenbeck, Chief Architect, Carol, Bohr" in the original document order as the result set.

Now, several enhancements to the system and method described above will be provided. In accordance with another aspect of the invention, the XMLName Table may be cached in memory. In particular, to facilitate construction of the NamePath values, we can store the contents of XMLName Table in a hash table which we keep resident in memory. This prevents the execution of multiple queries against the database to obtain all the necessary XMLName values. In accordance with yet another aspect of the invention, the XMLName values may be divided into NameSpaces. In particular, as the number of XMLName values increases, it may become necessary to divide the values into various namespaces to keep the lengths of the names short. XMLName values from namespaces relevant for working with a particular document can then be brought into the cache when necessary without having to bring the entire XMLNameTable into memory.

In accordance with yet another aspect of the invention, the system may use base-64 encoding. In particular, to reduce the amount of storage required for the XMLName, NamePath, and OrderPath tables in the relational database, we could consider using a Base-64 encoding scheme instead of alphanumeric strings. In accordance with the invention, it is also possible to add a DigitPath attribute as an adjunct attribute to OrderPath so that the system can ensure proper sorting of nodes while obviating the need for place-holding characters as the number of characters increases. For example, to sort the paths "1/10/2" and "1/2/3" properly, the system would have needed to encode the second as "1/-2/3". However, if we added "1/2/1" and "1/1/1" as DigitPaths and ordered the results by these before OrderPaths, then we would be able to do without the place-holding dashes.

In accordance with the invention, a ReverseNamePath attribute may be automatically generated to further improve the speed of queries. In particular, since it is possible to have an XML document that is an instance of a DTD sub-tree, we may need to evaluate an expression such as:

"Select NodeValue from DocNode Table where NamePath like '%/1/2/3'"

Since indexes built on NamePath generally do not help in the execution of such queries, we can improve performance by having a ReverseNamePath attribute constructed by reversing the order of the XMLNames in the path expression. Thus, in accordance with the invention, the above query would now read:

"Select NodeValue from DocNodeTable where ReverseNamePath like '3/2/1/%'"

In accordance with the invention, the system may include a transformation engine that converts XPath expressions into equivalent SQL statements involving NamePath and OrderPath attributes so that the converted queries would then be executed against the repository.

In summary, a system and method for assigning attributes to XML document nodes to facilitate their storage and indexing in relational databases and the subsequent retrieval and re-construction of pertinent nodes and fragments in original document order is provided. Since these queries are performed using relational database query engines, the speed of their execution is significantly faster than that using more exotic systems such as object-oriented databases. Furthermor his method is portable across all vendor platforms, and so can be deployed at client sites without additional investments in database software.

In accordance with the invention, the hierarchical relationships of XML documents are encoded so that the XML documents may be mapped to a set of relational tables. Once the mapping and encoding is completed, then searching and querying of the XML documents may be done by mapping any XML query language (which is well known) to SQL (also well known) automatically.

While the foregoing has been with reference to a particular embodiment of the invention, it will be appreciated by those skilled in the art that changes in this embodiment may be made without departing from the principles and spirit of the invention as set forth in the appended claims.

CLAIMS:

1. A computer system for manipulating an XML document using a relational database, comprising:

a converter that receives an XML document and generates a pre-determined set of relational database tables based on the XML document;

a database for storing the relational database table; and

a searcher for querying the generated relational database table in the database to locate content originally in the XML document that is now stored in the relational database table wherein the located content is returned to the user as a portion of an XML document.

2. The system of Claim 1, wherein the converter further comprises a software module that generates a unique name attribute for each node in the XML document.

3. The system of Claim 2, wherein the converter further comprises a software module that generates a path attribute for a particular node of the XML document wherein the path attribute comprises a list of the name attributes for the one or more nodes from the particular node to a root node of the XML document.

4. The system of Claim 3, wherein the converter further comprises a software module that generates an order attribute for the particular node, the order attribute comprising an enumerated order of the particular node from the root node to the particular node.

5. The system of Claim 4, wherein the converter further comprises a software module that generates a NodeValue attribute containing a value of the particular node.

6. The system of Claim 5, wherein the searcher further comprises a query generator that generates a query into the database to find a piece of information in the database
corresponding to information in a node of the XML document and a converter that converts the results of the query into portions of an XML document that are displayed to the user.

7. The system of Claim 2, wherein the name attribute for each node in the XML document is stored in a hash table so that the name attributes are retrieved from the hash table instead of the database.

8. The system of Claim 2, wherein the name attributes of the nodes of the XML document are divided into one or more categories so that related name attributes are grouped together.

9. The system of Claim 1, wherein the name attributes are encoded using base-64 encoding.

10. The system of Claim 3, wherein the converter further comprises a software module that generates a reverse path comprising the list of name attributes from the path attribute in reverse order.

11. The system of Claim 1, wherein the converter further comprises a transform engine that converts Xpath expressions in the XML document into SQL queries.

12. A computer system for storing an XML document using a relational database, comprising:

a converter that receives an XML document and generates a relational database table based on the XML document;

the converter further comprising a software module that generates a unique name attribute for each node in the XML document, a software module that generates a path attribute for a particular node of the XML document wherein the path attribute comprises a list of the name attributes for the one or more nodes from the particular node to a root node of the XML document, a software module that generates an order attribute for the particular node, the order attribute comprising an enumerated order of the particular node from the root node to the particular node, and a software module that generates a NodeValue attribute containing a value of the particular node.

13. A method for manipulating an XML document using a relational database, comprising:

generating a relational database table based on an XML document wherein the information about each node of the XML document is stored in a row of the table;

storing the relational database table in a database; and

querying the generated relational database table in the database to locate content originally in the XML document that is now stored in the relational database table wherein the located content is returned to the user as a portion of an XML document.

14. The method of Claim 13, wherein generating the table further comprises generating a unique name attribute for each node in the XML document.

15. The method of Claim 14, wherein generating the table further comprises generating a path attribute for a particular node of the XML document wherein the path attribute comprises a list of the name attributes for the one or more nodes from the particular node to a root node of the XML document.

16. The method of Claim 15, wherein generating the table further comprises generating an order attribute for the particular node, the order attribute comprising an enumerated order of the particular node from the root node to the particular node.

17. The method of Claim 16, wherein generating the table further comprises generating a NodeValue attribute containing a value of the particular node.

18. The method of Claim 17, wherein querying the database further comprises generating a query into the database to find a piece of information in the database corresponding to information in a node of the XML document and converting the results of the query into portions of an XML document that are displayed to the user.

19. The method of Claim 14 further comprising retrieving the name attribute for each node in the XML document from a hash table so that the name attributes are retrieved from the hash table instead of the database.

20. The method of Claim 14, wherein the name attributes of the nodes of the XML document are divided into one or more categories so that related name attributes are grouped together.

21. The method of Claim 13, wherein the name attributes are encoded using base-64 encoding.

22. The method of Claim 15, wherein generating the table further comprises generating a reverse path comprising the list of name attributes from the path attribute in reverse order.

23. The method of Claim 13, wherein generating the table further comprises converting Xpath expressions in the XML document into SQL queries.

24. A data structure that stores a node of interest of an XML document in a relational database, the data structure comprising:

an XMLName attribute comprising a unique name for the node of interest;

a NamePath attribute comprising a list of the XMLName attributes for the one or more nodes from the node of interest to a root node of the XML document;

an OrderPath attribute comprising an enumerated order of the node of interest from the root node to the node of interest; and

a NodeValue attribute containing a value of the node of interest.

25. The data structure of Claim 24, wherein the data structure comprises a table in a relational database and each attribute comprises a column in the table in the relational database.

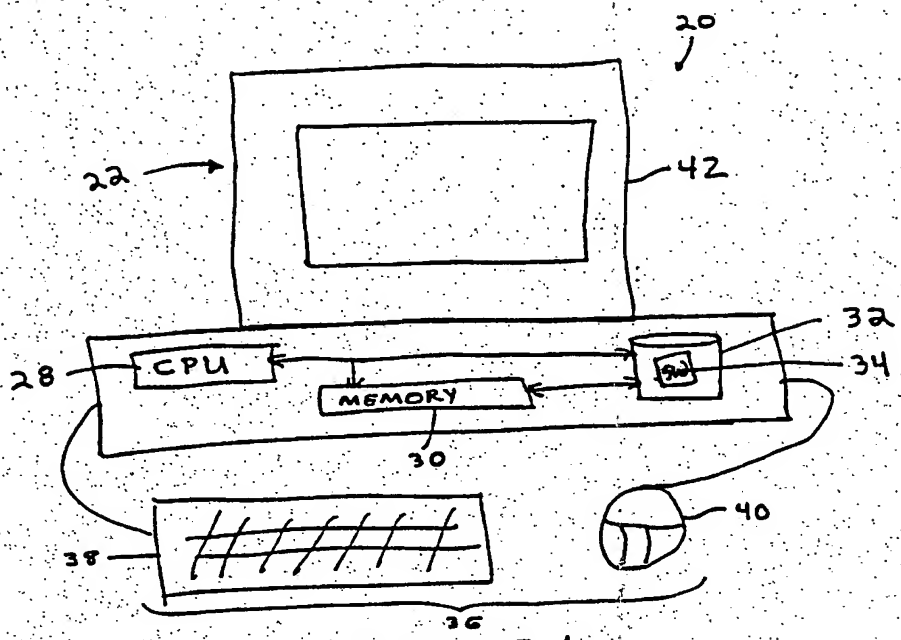


FIGURE 1

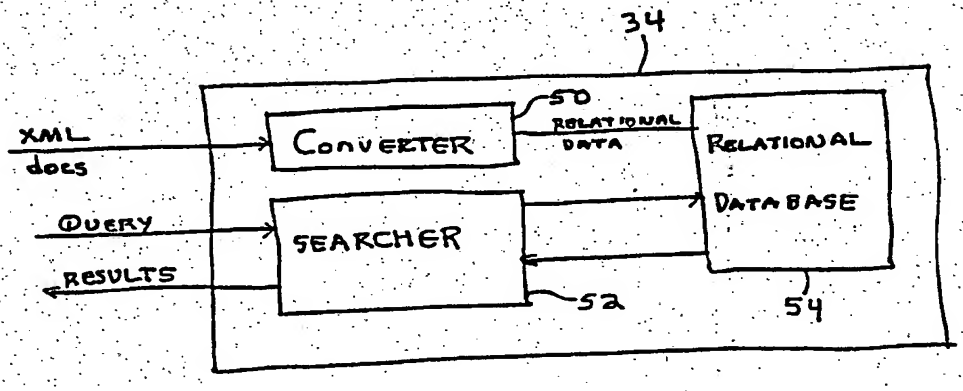


FIGURE 2

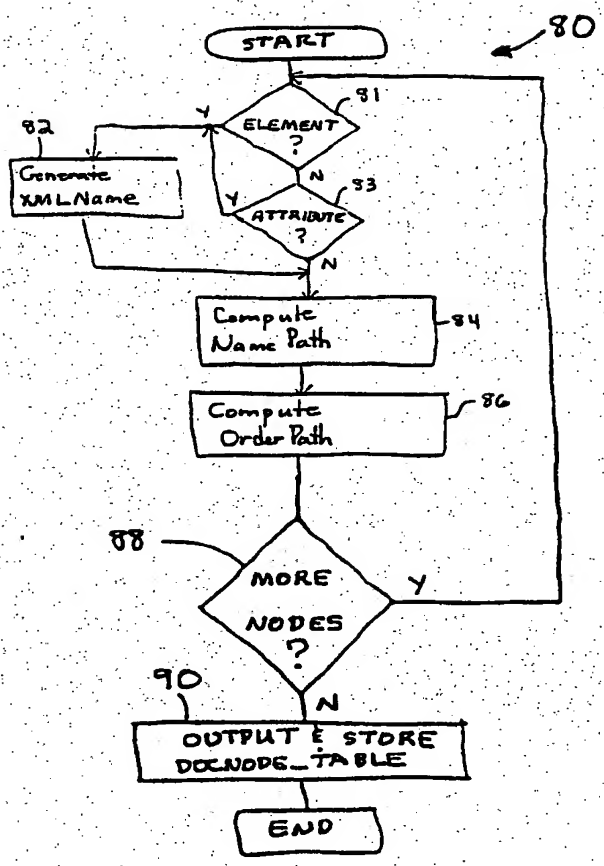


FIGURE 5

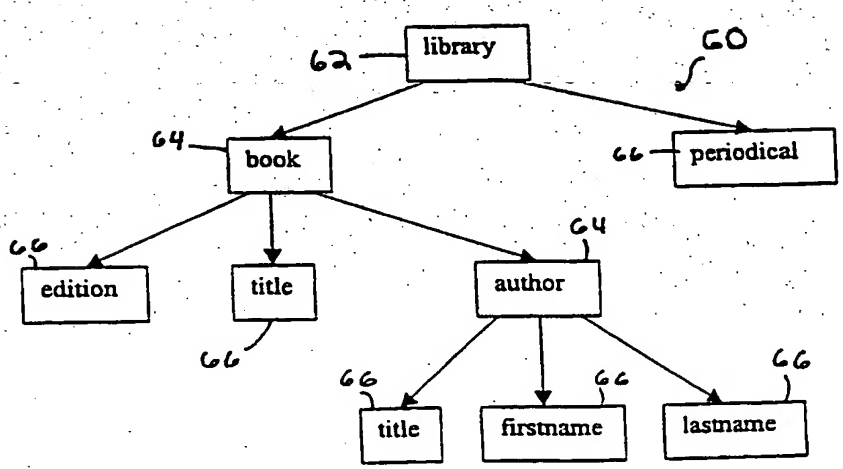


FIGURE 3

```
<library>
    <book edition='first'>
        <title>The XML Revolution</title>
        <author>
            <title>Software Engineer</title>
            <firstname>David</firstname>
            <lastname>Hollenbeck</lastname>
        </author>
        <author>
            <title>Chief Architect</title>
            <firstname>Carol</firstname>
            <lastname>Bohr</lastname>
        </author>
    </book>
    <book edition='second'>
        <title>Java Classes for XML</title>
        <author>
            <firstname>Carol</firstname>
            <lastname>Hollenbeck</lastname>
        </author>
        <author>
            <title>XML Guru</title>
            <firstname>David</firstname>
            <lastname>Bohr</lastname>
        </author>
    </book>
</library>
```

70

FIGURE 4

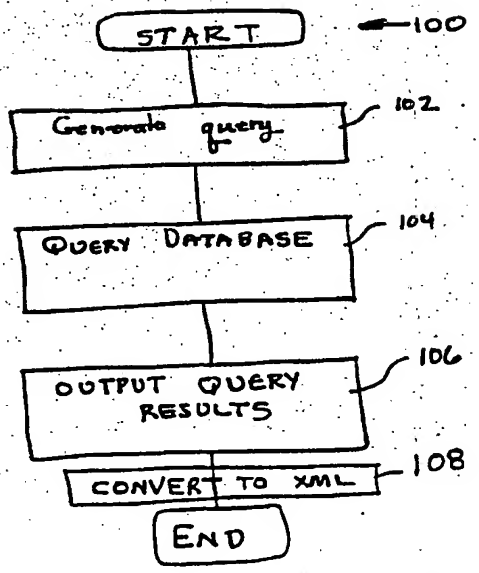


FIGURE 6

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 June 2001 (14.06.2001)

PCT

(10) International Publication Number
WO 01/42881 A3

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US00/42665

(22) International Filing Date: 6 December 2000 (06.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/169,101 6 December 1999 (06.12.1999) US
Not furnished 5 December 2000 (05.12.2000) US

(71) Applicant: B-BOP ASSOCIATES, INC. [US/US]; Suite 100, One Bay Plaza, 1350 Old Bayshore Highway, Burlingame, CA 94010 (US).

(72) Inventors: DODDS, David; 16 Old Barn Road, Stamford, CT 06905 (US). KUO, Larry; 120 Morning Star Drive, San Jose, CA 95131 (US). SENGUPTA, Soumitra; 15 First Street, Apt. 5, Stamford, CT 06905 (US). LINDSEY, Bill; 2203 Hastings Drive, Apt. 28, Belmont, CA 94002 (US).

(74) Agent: LOHSE, Timothy, W.; Gray Cary Ware & Freidenrich LLP, 400 Hamilton Avenue, Palo Alto, CA 94301-1825 (US).

(81) Designated States (national): AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

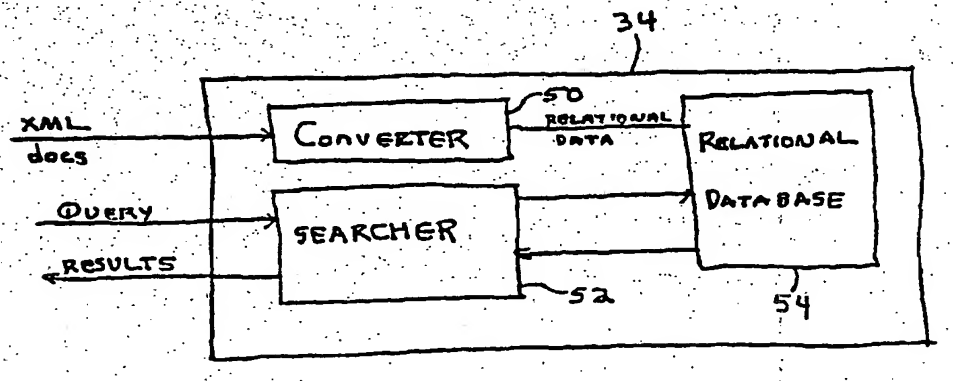
(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— with international search report

(88) Date of publication of the international search report:
10 January 2002

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR THE STORAGE, INDEXING AND RETRIEVAL OF XML DOCUMENTS USING RELATIONAL DATABASES



(57) Abstract: A system (34) and method for assigning attributes to XML documents to facilitate their storage and retrieval in relational databases (54). A converter (50) accepts XML documents, processes them and outputs relational data about the XML documents which is stored in the relational database (54). A searcher (52) using SQL query engines performs queries to retrieve the documents and sends the results of the query in XML form to the users.

WO 01/42881 A3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/42665

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/30
US CL : 707/3, 101

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/3, 101, 10, 102, 103, 104

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST, IEL

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A, P	US 6,154,738 A (CALL) 28 November 2000 (28.11.2000), ALL.	1-23
A, P	US 6,125,391 A (MELTZER et al) 26 September 2000 (26.09.2000), ALL.	1-23

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
02 May 2001 (02.05.2001)

Date of mailing of the international search report
25 MAY 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Uyen T Le
Telephone No. 305-9000

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/42665

Box I Observations where certain claims were found unsearchable (Continuation of Item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☒ Claim Nos.: 24 and 25
because they relate to subject matter not required to be searched by this Authority, namely:
Claims 24, 25 recite pure descriptive non-functional subject matter.

2. ☐ Claim Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claim Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of Item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest
☐ The additional search fees were accompanied by the applicant's protest.
☐ No protest accompanied the payment of additional search fees.